Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 5, May 2015, pg.496 – 505

RESEARCH ARTICLE

Practical Appraisal of Distinguish Active Queue Management Algorithms

Nancy[1], Gurpreet Singh[2]

[1]M.Tech. Scholar, Department of Computer Science and Engineering, Yamuna Institute of Engineering and Technology, Gadholi, Haryana, 133103, India

[2]Dean Academics, Head in the Department of Computer Science and Engineering, Yamuna Institute of Engineering and Technology, Gadholi, Haryana, 133103, India

[1] nancy222.arora@gmail.com; [2] gps_ynr@yahoo.com

Abstract*— The internet and its applications are an integral part of our daily life. Today's Internet needs to provide Best Effort Service. With the rapid transformation of the Internet into a commercial infrastructure, demands for quality of service have rapidly developed. Traffic is processed as quickly as possible, but there is no guarantee of timelines or actual delivery. The people of the modern world are very much dependent on various network services like VOIP, Video Conferencing and File Transfer. So to prevent the problem of congestion control and synchronization, various active queue management (AQM) techniques are used. AQM algorithms execute on network routers and detect initial congestion by monitoring some functions. When congestion occurs on the link the AQM algorithm detects and provides signals to the end systems. The focus of this work is to study the behaviors of varied queue managements, including Drop-Tail, RED (Random Early Detection), REM (Random Exponential Marking), Blue and SFQ (Stochastic Fair Queuing). The performance metrics are used for comparison between these AQM techniques such as average throughput, number of packets lost, delivery ratio, average end to end delay and average jitter. The simulation is done using the network simulator (NS-2).

***Keywords*— Active Queue Management, Congestion Control, Drop-Tail, RED, REM, Blue, SFQ, Throughput**

I. INTRODUCTION

Over the last decade, the flow and congestion control mechanisms [10] of TCP have been used to adaptively control the rates of individual connections sharing IP network links. However, TCP congestion control algorithm over current drop-tail networks has one serious drawback, i.e. TCP source reduces its transmission rate only after losing packets. This is an important error since considerable time may pass between the packet drop at the router and its detection at the source. In the meantime, a large number of packets may be dropped as the sender continues to transmit at a rate that the network cannot support. Therefore, even with techniques such as congestion avoidance, slow start, fast retransmit and fast recovery mechanism [9], the performance of the TCP congestion control algorithm over current drop-tail networks is inadequate in a heavily loaded network.

Hence, the most effective detection of congestion can occur in the router itself. The router can reliably distinguish between propagation delay and persistent queuing delay. Moreover, the router [14] has a unified view of the queuing behavior over time and the decisions about the duration and magnitude of transient congestion that can be allowed at the gateway are best made by the router itself.

This leads many researches on the dynamic queue managements. The basic idea for dynamic queue management is to implicitly or explicitly notify sources to decrease the transmission rates before the queue overflows in hope that this coordination between sources and network will eliminate any future sustained packet loss. In order to encounter the increasing packet loss rates caused by an exponential increase in network traffic, the Internet Engineering Task Force (IETF) has considered the deployment of active queue management techniques (AQM)[13]. The basic idea behind an active queue management algorithm is to convey congestion notification early to the TCP [12] endpoints so that they can reduce their transmission rates before queue overflow and sustained packet loss occur. A typical dynamic queue management is RED (Random Early Detection), which was recommended by the IETF for deployment in IP routers/networks and is supported by many routers. It is now widely believed that a RED-controlled queue performs better than a drop-tail queue. However, the inherent design of RED [7], which uses the available queue length as the indicator of the severity of congestion, makes it difficult to parameterize RED queues to give a good performance under different network scenarios. A contrast to RED, Blue [1] use packet loss and link idle events to manage the congestion. The basic idea of Blue is to maintain a single probability to mark packets when they are queued. The probability is changed according the utilization ratio of the link.

II. ACTIVE QUEUE MANAGEMENT TECHNIQUES

The TCP [16] detects congestion only after a packet has been dropped from the queue (according to the drop tail-algorithm). However, it is clearly undesirable to have large queues that are full most of the time, since this will significantly increase the delays. Therefore and also keeping in mind the ever increasing speed of networks, it is ever more important to have a mechanism that keeps the overall throughput high, but at the same time keeps the average queue size as low as possible. Note that in order to maximize the network throughput [8], queues should not necessarily be kept completely empty all the time, since this will result in under-utilization of the link, but in order to have a small queuing delay, the queue length should be kept sufficiently small.

In order to fulfill the above needs, a wide range of AQM algorithms [4] have been proposed. The purpose of these algorithms is to provide a mechanism to detect network congestion early and to start dropping packets from the router queue before this congestion will affect the network throughput too much. The definition of too much depends on the Quality of Service (QoS) to be delivered by the network.

AQM algorithms have been designed for implementation at network routers, as opposed to implementation at end nodes, such as TCP sender or receiver entities. This choice is advocated by the fact that detection of congestion can be carried out more effectively in the router itself. A network router can reliably distinguish between propagation delay and persistent queuing delay. Only the router has a unified view of the queuing behaviour over time; decisions about the duration and magnitude of congestion to be allowed at the router are therefore best made by the router itself. The AQM algorithm [11] controls the arrival rate of packets into the queue, by Explicit Congestion Notification (ECN) marking or packet dropping to generate the congestion signal that controls the source rate.

A. Drop-Tail

Drop Tail [11] is a simple active queue management (AQM) technique used in many routers. It doesn't differentiate traffic from different sources. As long as the queue is filled up, it will drop subsequent packets arrived. In other words, drop the tail of a sequence of packets.

Drop Tail is simple and easy to implement, however, it suffers from a couple of drawbacks.

● It doesn't distribute buffer space fairly. Because Drop Tail doesn't differentiate traffic from different sources, sources with higher traffic volume will take more buffer space.

● If multiple TCP connections exist in the system and a buffer overflow will cause TCP global synchronization, which reduce the network throughput and utility significantly.

B. RED (Random Early Detection)

Random Early Detection (RED) [11] was first proposed AQM mechanism and is also promoted by the Internet Engineering Task Force (IETF) [2]. Random Early Detection (RED) was introduced in 1993 [3] by Floyd and Jacobson. RED provides congestion avoidance by controlling the queue size at the gateway [3]. RED is customized for TCP connection across IP routers, it's considered to avoid congestion. It notifies the cause before the congestion truly happens rather than wait till it actually occurs. It provides a method for the gateway to provide some feedback to the resource on congestion status. In order to solve the problem of passive queue management technique this section proposes and evaluates a primarily different queue management algorithm called RED. RED is designed to be used in conjunction with TCP, which currently detects congestion by means of timeouts (or some other means of detecting packet loss such as duplicate ACKs) [10]. RED has been designed with the objective to:

- Reduce packet loss and queuing wait,
- Avoid global synchronization of sources
- Maintain high link utilization, and
- Remove biases against bursty sources

RED algorithm monitors the average queue size and marks packets. If the buffer is almost empty, all incoming packets are accepted. As the queue grows, the probability of dropping an incoming packet grows too. When the buffer is full, all incoming packets are dropped [3]. Thus RED buffer mechanism works with constant bit rate (CBR) traffic can be used at an early stage to know the effect of change of network parameters over system performance. The main aim of RED is to control the queue size and indicating the end hosts when to slow down their packet transmission rate. It takes benefit of the congestion control mechanism of TCP by randomly dropping packets earlier to periods of high congestion, RED [7] tells the packet source to reduce its transmission rate. Assuming the packet source is using TCP, it will reduce its transmission rate until all the packets reach their destination, representing that the congestion is cleared.

Random early detection is a queue management scheme that is proposed to respond the shortcomings of drop tail. RED [3] perfectly notifies one of the sources of congestion by randomly dropping an arriving packet. The selected source is informed of the packet loss and its sending rate is reduced accordingly. Therefore, congestion is alleviated. It is an early congestion declaration. The dropping probability is a function of average queue length. When the queue tenure grows, congestion builds up. Then, the dropping probability increases in order to supply enough early congestion notifications another goal of RED [4] is to eradicate biases against busty sources in the network. This is done by limiting the queue, use so that there is always room left in the queue to buffer transient bursts. In addition, the marking purpose of RED takes into account the last packet marking time in its calculations in order to reduce the probability that successive packets belonging to the same burst are marked.

C. REM(Random Exponential Marking)

Random Exponential Marking (REM) [8] is an active queue management scheme that measures congestion not by performance measure such as loss or delay, but by quantity. REM can achieve high utilization, small queue length, and low buffer overflow probability. Many works have used control theory to provide the stable condition of REM without considering the feedback delay. In case of (Random Exponential Marking) REM, the key idea is to decouple congestion measure from performance measure (loss, queue length or delay). In REM, the user rates are matched by clearing buffers irrespective of number of users. The sum of link prices, summed over all the routers in the path of the user to the end to end marking [9].

Random Exponential Marking (REM) has the following key features:

- Match rate clear buffer: It attempts to match user rates to network capacity while clearing buffers (or stabilize queues around a small target), regardless of the number of users.
- Sum prices: The End-to-end marking (or dropping) probability observed by a user depends in a simple and precise manner on the sum of link prices (congestion measures), summed over all the routers in the path of the user.

The first feature implies that, contrary to the conventional wisdom, high utilization is not achieved by keeping large back- logs in the network, but by feeding back the right information for users to set their rates. We present simulation results which demonstrate that REM can maintain high utilization with negligible loss or queuing delay as the number of users increases.
The second feature is essential in a network where users typically go through multiple congested links. It clarifies the meaning of the congestion information embedded in the end- to-end marking (or dropping) probability observed by a user, and thus can be used to design its rate adaptation.

D. Blue

One of the fundamental problems with RED [7] and other active queue management techniques is that they rely on queue length as an estimator of congestion. But this approach has an inherent problem in determining the severity of congestion. For instance, when a large number of TCP [12] sources are active, the aggregate traffic generated is extremely bursty. Bursty traffic often defeats the active queue management techniques used by RED since queue lengths grow and shrink rapidly well before RED can react. Even though RED can achieve an ideal operating point, it can only do so when it has a sufficient amount of buffer space and is correctly parameterized.
The key idea behind Blue is to perform queue management based directly on packet loss and link utilization rather than on the instantaneous or average queue lengths. This is in contrast to all known active queue management schemes which use some form of queue occupancy in their congestion management. Blue [1] maintains a single probability that it uses to mark (or drop) packets when they are enqueued. If the queue is continually dropping packets due to buffer overflow, Blue incrementprobability value, thus increasing the rate at which it sends back congestion notification. Conversely, if the queue becomes empty or if the link is idle, BLUE decreases its marking probability. This effectively allows Blue [10] to "learn" the correct rate it needs to send back congestion notification.

E. SFQ (Stochastic Fair Queuing)

Stochastic Fairness Queueing (SFQ) is a simple implementation of the fair queueing algorithms family. It is less accurate than others, but it also requires fewer calculations while being almost perfectly fair. The key word in SFQ is conversation (or flow), which mostly corresponds to a TCP session or a UDP stream. Traffic is divided into a pretty large number of FIFO queues, one for each conversation. Traffic is then sent in a round robin fashion, "giving each session the chance to send data in turn". "This leads to very fair behaviour and disallows any single conversation from drowning out the rest". SFQ [24] is called "Stochastic" because it does not really allocate a queue for each session, it has an algorithm which divides traffic over a limited number of queues using a hashing algorithm. Because of the hash, multiple sessions might end up in the same bucket, which would halve each session is chance of sending a packet, thus halving the effective speed available. To prevent this situation from becoming noticeable, SFQ changes its hashing algorithm quite often so that any two colliding sessions will only do so for a small number of seconds.

III. SIMULATION AND COMPARISON

In this section, we will compare the performances of various AQM techniques such as Drop-Tail, RED, REM, Blue and SFQ. Our simulation is based on NS-2. In order to evaluate the performance of Drop-Tail, RED, REM, Blue and SFQ, a number of simulation experiments were run using NS-2 over a small network shown in Fig. 1 with a number of input nodes. The main parameters that are taken into consideration are Average throughput, Number of packets lost, Delivery Ratio, Average End to End Delay and Average Jitter. The comparative investigation on different queuing disciplines based on heavy congestion is evaluated.

A. SIMULATION SETTINGS

As different algorithms have different preferences or assumptions for the network configuration and traffic pattern, one of the challenges in designing our simulation is to select a typical set of network topology and parameters (Average throughput, Number of packets, Number of packets lost, Delivery Ratio, Average End to End Delay and Average Jitter) as the basis for evaluation. TCP (FTP application in particular) and UDP flows (CBR application in particular) are chosen as typical traffic patterns.

In our simulation, we use two TCP flows and one UDP flow. The bottleneck link in this scenario is the link between 3 and node 4 as shown in fig.1.This paper shows a wired topology of eight nodes shown in the fig.1. Nodes labeled 0,1 and 2 are connected to node 3 via a duplex link and further node 3 is connected to node 4 and node 4 is connected to node 5 and node 6 with two way open communication channel, node 5 and node 6 are also connected with node 7 via duplex link. TCP agents of different variations are attached with node 0 and node 1 and send the FTP data to the node 3.A TCP sink is being attached to node 3 which will receive the packets send by node 0 and node 1. A UDP agent is also attached to node 2 and sends the Telnet packets towards the node 3. To receive the packets send by node 2, a null agent is attached to node 3. Both the TCP as well as UDP packets are routed via a single common sub-path (3-4). Both types of packets are queued at node 3. We discussed the results of the simulated scenario of various Active Queue Management algorithms as Drop-Tail, RED(Random Early Drop), REM(Random Exponential Drop), Blue, SFQ(Stochastic Fair Queuing) with different parameters using NS-2 simulator and compare their results.

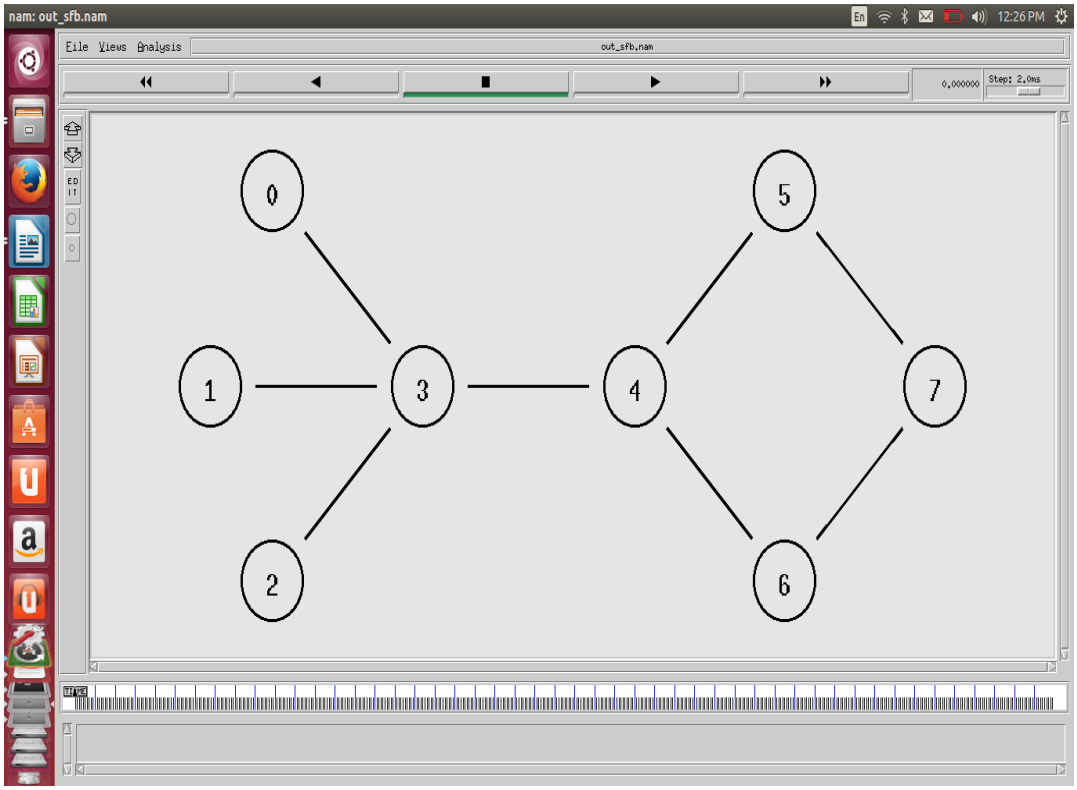

Fig. 1: Topology for implementation of various Active Queue Management Techniques

In this paper, we consider the different parameters for comparing various AQM Algorithms such as throughput, number of packet loss, delivery ratio, end to end delay and jitter. We have conducted experiments on the given topology in this paper and analyzed the results of various AQM algorithms on the basis of selected parameters. The values generated in the experiment are given below in the tabular form as shown in table 1.

TABLE 1

TABULAR REPRESENTATION OF AQM TECHNIQUES

Algorithm	Drop Tail	RED	REM	Blue	SFQ
Average throughput	3552.274243	3192.378125	3182.999535	2658.272117	2903.290146
Number of packets lost	43	38	45	48	31
Delivery Ratio	99.92619926	99.92726716	99.91377824	99.89009983	99.93497913
Average End to End Delay	0.166048816	0.103720936	0.129191748	0.142926659	0.108191092
Average Jitter	0.000976774	0.000789374	0.001104317	0.001347498	0.00031382

Throughput: Throughput is defined as the total amount of data received by destination node from the source node divided by the total time it takes from the destination to get the last packet and it is normally measured in bits per second (bit/s or bps).However, we have calculated it in packets per second. This is the main performance measure characteristic, and most widely used. This measure how soon the ability to get a certain amount of data send by the sender. Throughput is an important factor which directly impacts the network performance.

Average Throughput= Total number of packets send / Overall time.

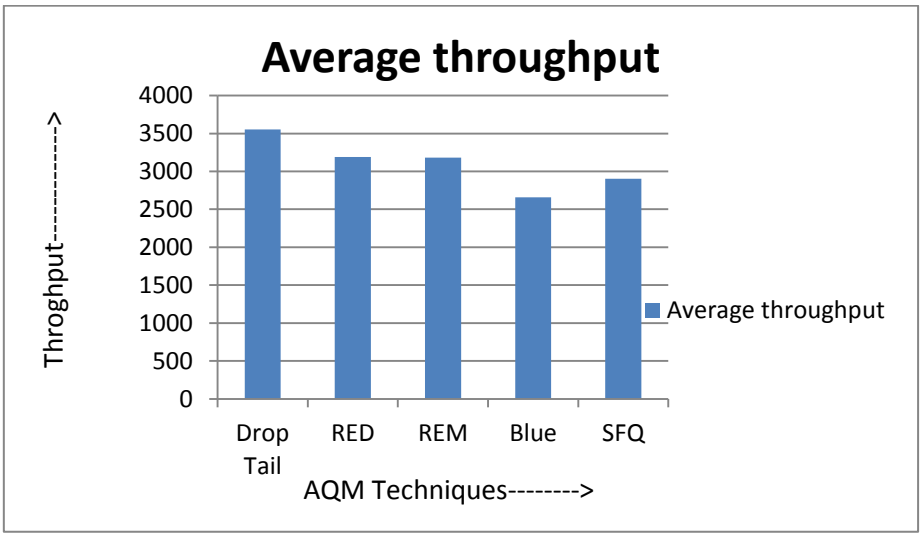

Fig. 2: Throughput

It has been observed that Drop-Tail had a best throughput and Blue had least throughput among all these algorithms for the simulation. It could be observed one point on the throughput graph whenever smooth growth in throughput has been broken. It indicated about a starting point when dropping of packet took place. This achieved point in each algorithm has a same ratio as compared to their maximum achieved throughput.

1) *Packet Loss*: Packets can be lost in a network because they may be dropped when a queue in the network node overflows. The amount of packet loss during the steady state is another important property of a congestion control scheme.

The larger the value of packet loss, the more difficult it is for transport layer protocols to maintain high bandwidths, the sensitivity to loss of individual packets, as well as frequency and patterns of loss among longer packet sequences is strongly dependent on the application itself. This characteristic can be specified in a number of different ways, including loss rate, loss patterns, loss free seconds, and conditional loss probability.

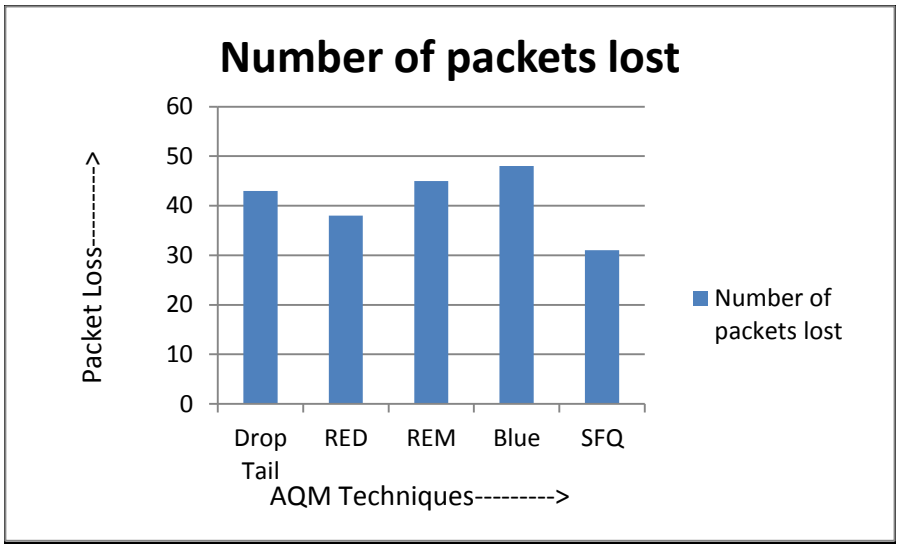

Fig. 3: Packet loss

Fig. 3 shows the number of packets lost during transmission of data from the source to the destination. From fig. 3, it is clear that the data rate of SFQ is better than other techniques.

2) *Delivery Ratio*: In TCP the Packet delivery ratio is the ratio of total packets sent by the source node to the successfully received packets by the destination node.

Delivery Ratio= Total data packet delivered successfully x 100 /data packet generated

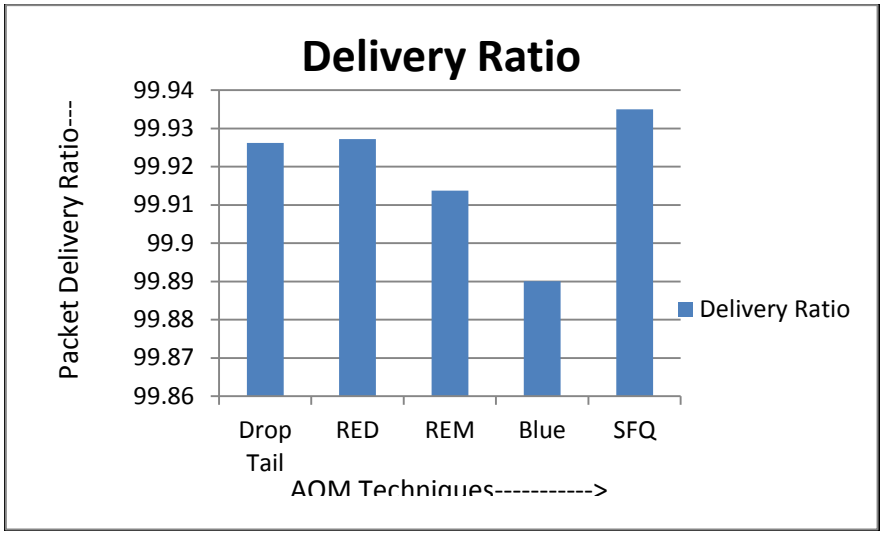

Fig. 4: Delivery Ratio

Fig 4. shows the delivery ratio of TCP in terms of percentage. This shows that SFQ has better delivery ratio than other variants.

3) *Jitter:* The time difference in packet inter-arrival time to their destination can be called jitter. Jitter is specific issue that normally exists in network.

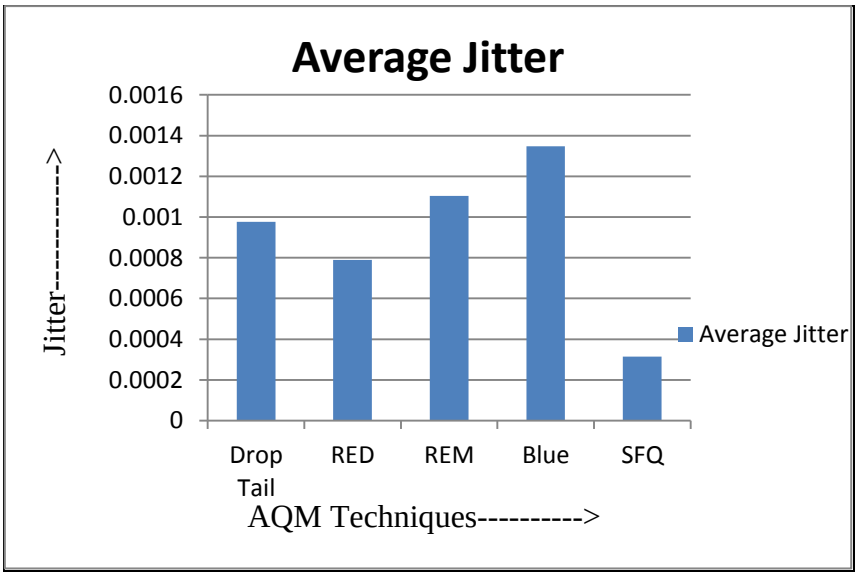

Fig. 5: Jitter

Fig. 5 shows the time difference in inter arrival time to their destination. SFQ has lowest variations in inter arrival time among all the TCP variants which we discussed here.

4) Average Delay: Delay is the time elapsed while a packet travels from one point (e.g., source premise or network ingress) to another (e.g., destination premise or network degrees). The larger the value of delay, the more difficult it is for transport layer protocols to maintain high bandwidths. This characteristic can be specified in a number of different ways, including average delay, variance of delay (jitter), and delay bound. In this paper, we calculated end to end delay.

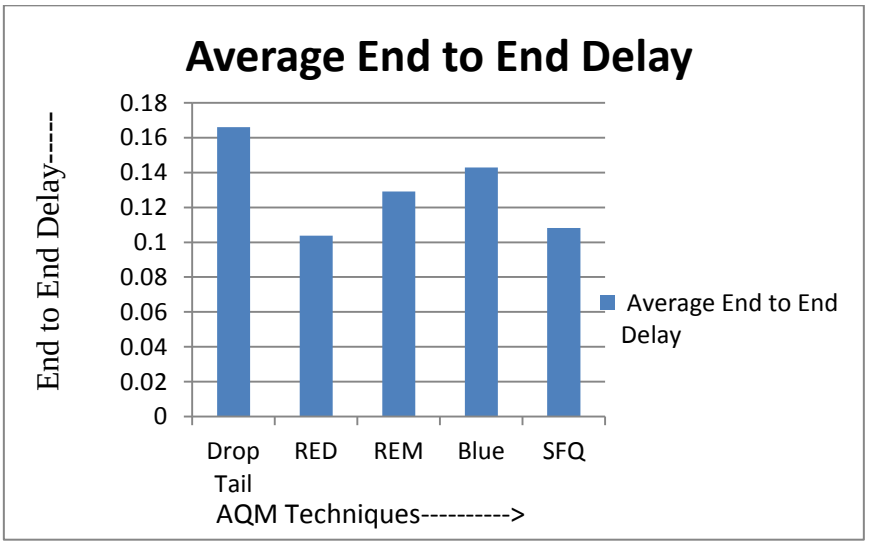

Fig. 6: End to End delay

Fig 6. plots the actual response time for each packet achieved in Drop-Tail, RED, REM, Blue, SFQ. This shows that the RED has less delay as compared to other variants.

IV. CONCLUSION

In this paper, we have made an effort to understand various popular queue management techniques and compare them using various parameters. Through this paper, we tried to understand the queue management techniques over the traffic loaded network. As our results show that, not a single algorithm is self-sufficient. We have calculated Average throughput, Number of packets lost, Delivery Ratio, Average End to End Delay and Average Jitter for the given topology. Our results show that the Drop-Tail algorithm has maximum throughput while Blue has minimum throughput. Also Drop-Tail has achieved maximum end-to-end delay and RED got minimum end-to-end delay. SFQ has minimum packets dropped, while Blue has the highest packets being dropped or lost in the network. While RED is an intermediate in terms of lost packets. But, simultaneously, if we consider Average Jitter, SFQ achieved the best results. Our simulation results conclude that not a single queue management technique is sufficient in terms of all the parameters. There are many parameters of network performance measurement. But if we consider our chosen parameters, SFQ has shown somewhat greater impact as compared to other AQM techniques.

REFERENCES

[1]. W. Feng et. al. , Blue: A new class of Active Queue Management Algorithms, *Technical Report CSE-TR- 387-99*, Dept. EECS, Univ. MI, Apr. 1999.
[2]. B. Braden, D. Clark, J. Crow Croft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclaw ski and L. Zhang RFC 2309: Recommendations on Queue Management in April 1998
[3]. S. Floyd and V. Jacobson, "Random early detection gateway for Congestion avoidance", IEEE/ACM Transaction on Networking, vol. 1, no. 4, pp. 397-413, Aug. 1993.
[4]. Saurabh Sarkar, Mrs. Geeta Sikka, Ashish Kumar, "Classification of Active Queue Management Algorithms for Application Specific Traffics", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 5, ISSN: 2277 128X, May 2012.
[5]. S. Floyd, R. Gummdi, and S. Shenker, "Adaptive RED, An algorithm for increasing the robustness of RED", AT&T Center for Internet Research at ICSI, Aug. 2001.
[6]. Mahmud Etbega Mohamed, M.E. Woodward, "Adaptive Early Random Drop: An algorithm for controlling queueing delay in a buffer", Proceeding of 16th Telecommunication Forum IEEE, Belgrade-Serbia, 2008.
[7]. Neha, Abhinav Bhandari," RED: A high link utilization and fair algorithm", International Journal of Computer Applications Technology and Research,Volume 3, Issue 7, 415 - 419, 2014, ISSN: 2319–8656

[8]. Sanjeev Patel, P. K. Gupta, Arjun Garg, Prateek Mehrotraand Manish Chhabra, "Comparative Analysis of Congestion Control Algorithms Using ns-2", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5, No 1, September 2011

[9]. T. Bonald, M. May, and J. Bolot. "Analytic evaluation of RED performance.", in INFOCOM, Vol. 3, pp 1415–1424, 2000.

[10]. W. Feng, "Improving Internet Congestion Control and Queue Management Algorithms", PhD Thesis, University of Michigan, 1999.

[11]. Dr. T. Bhaskar Reddy, Ali Ahammed, and Reshma banu, "Performance Comparison of Active Queue Management Techniques", IJCSNS Vol.9, Issue 2, pp405-408, February 2009.

[12]. Amanpreet Kaur, Gurpreet Singh, Baninder Singh, "Evaluation of Congestion control variants of TCP by strolling propagation delay in NS-2", International Journal for Applied Engineering and Research (Vol. V) with print ISSN No: 0973-4562 and Online ISSN No: 1087—1090, April, 2011.

[13]. Santhi, V., Natarajan, A. M, "Performance Analysis of Active Queue Management Algorithms", International Journal on Information Sciences and Computing, Vol.3, Issue 1, January 2009.

[14]. G.F.Ali Ahmed, Reshma Banu, Analyzing the performance of Active Queue Management Algorithms, International journal of Computer Networks & Communications (IJCNC), Vol.2, Issue 2, March 2010.

[15]. Amanpreet Kaur, Gurpreet Singh, Deepti Chauhan, "Radical Analysis of TCP Alternatives: Tahoe, New Reno, Sack, Vegas on the basis of imitation in NS-2", in the proceedings of Journal of Yamuna Journal of Technology and Management, Vol 1, Issue 1, Nov, 2011.

[16]. W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "Techniques for eliminating packet loss in congested TCP/IP networks", Univ. Michigan, Ann Arbor, MI, Tech. Rep.UMCSE-TR-349-97, Oct. 1997.

[17]. Vandana Khare, Dr. Y. Madhavee Latha, Dr. SrinivasRao, "Comparative Analysis of Queuing Mechanism for WCDMA Networks",International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 2, Issue 2, March –April 2013.

[18]. Bernhard Suter, T. V. Lakshman, Dimitrios Stilidis, and Abhijit Choudhury, "Efficient Active Queue Management for Internet Routers", November 1997, Interop '98

[19]. L. Le, J. Aikat, K. Jeffay, and F.D. Smith, "The Effects of Active Queue Management and Explicit Congestion Notification on Web Performance", IEEE/ACM Transactions on Networking, Volume 15, Issue 6, Pages 1217-1230, December 2007.

[20]. Bartek Peter Wydrowski, "Techniques in Internet Congestion Control", Ph.D. Thesis, Electrical and Electronics Engineering Department, The University of Melbourne, Febuary 2003.

[21]. L. Brakmo and S. O'Malley, "TCP-Vegas: New techniques for congestion detection and avoidance", In ACM SIGCOMM'94, pp. 24-35, OCT, 1994.

[22]. Kamalpreet Kaur, Navdeep Kaur, Gurjeevan Singh, "Performance comparison of queuing algorithms: a review", IOSR Journal of Electronics and Communication Engineering (IOSR-JECE),e-ISSN: 2278-2834, p-ISSN: 2278-8735.Volume 8, Issue 4, PP 46-48, Nov. -Dec. 2013.

[23]. Wu Chung Feng., Kang G.Shin., Dilip D.Kandlur., Debanjan Saha .,"The BLUE Active Queue Management Algorithms" IEEE ACM Transactions on Networking, August 2002.

[24]. Nancy, Gurpreet Singh, "Comparing Different Active Queue Management Techniques", International Journal of Emerging Research in Management &Technology, ISSN: 2278-9359 (Volume-4, Issue-5), PP 50-56, May 2015.

